**文档信息及修订记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 项目名称 | 启明星V10 | | | | |
| 项目编号 |  | | 项目负责人 |  | |
| 文档编号 |  | | 文档密级 |  | |
| 文档主送 |  | | | | |
| 文档抄送 |  | | | | |
| 修 订 人 | 修订日期 | 修订说明 | | | 版本号 |
| 郝林 | 2020/11/30 | 创建并添加基本内容。 | | | V1.0.0.1 |
| 李霖 | 2021/3/17 | 新增昨持仓快照响应（详见5.3.8），修改实时持仓快照响应数据结构（详见5.3.9）。 | | | V1.0.0.2 |
| 李霖 | 2021/4/16 | 添加windows版本支持。 | | | V1.0.0.3 |
| 李霖 | 2021/5/09 | 1. 添加手续费参数响应(详见5.3.7)； 2. 添加保证金参数响应(详见5.3.8)； 3. 添加市场状态信息响应(详见5.3.9)及通知(详见5.3.28)； 4. 添加浮盈通知（详见5.3.29）； 5. 更新UDP协议，使用方法详见样例章节10; 6. 更新撤单请求使用方法（详见10.2）; 7. 协议变更：**委托号和成交号类型变更为整型**，其他变更详见章节11。 | | | V1.0.1.0 |
| 李霖 | 2021/5/31 | 1. 修改部分函数接口返回值类型(见5.2.3、5.2.4、5.2.5、5.2.6、5.2.8); 2. 更新UDP协议，使用方法详见样例章节10; 3. 协议变更：**浮盈通知添加流号字段**，其他变更详见章节11。 | | | V1.0.1.1 |
| 李霖 | 2021/6/23 | 日志功能优化。   1. 不绑定日志线程CPU将不会创建日志; 2. 同一个自然日只产生一份任务（log）日志,每创建一个API实例产生一份数据(data)日志； 3. 释放API实例同时释放日志句柄。 | | | V1.0.1.2 |
| 李霖 | 2021/7/5 | 修复不绑定日志线程CPU也会创建日志线程的问题。 | | | V1.0.1.3 |
| 李霖 | 2021/8/27 | 1）添加组合报单样例，见章节10.2；  2）协议更新：添加**组合报单请求**协议等，见章节11中的表4； | | | V1.0.1.4 |
| 李霖 | 2021/10/18 | 1. 添加TCP报撤单功能，见章节5.2.11~5.2.14； 2. 添加席位信息通知，见章节5.3.30； 3. 登录应答、席位信息协议变更，见章节11，表5。 | | | V1.0.1.6 |
| 李霖 | 2021/11/5 | 1. 添加用户账户密码修改功能。 | | | V1.0.1.7 |
| 李霖 | 2021/12/10 | 1. 心跳功能进行优化，对其他功能无影响。 | | | V1.0.1.8 |
| 李霖 | 2022/2/21 | 1. 硬件信息采集功能优化，对其他功能无影响。API初始化时将会主动进行信息采集，用户不需要再调用GetSystemInfo函数。 | | | V1.0.1.9 |
| 李霖 | 2022/4/13 | 1. 添加交易权限查询和推动功能，见章节5.3.10、5.3.35和5.3.36； 2. 添加客户品种白名单查询功能，见章节5.3.11。 | | | V1.0.1.10 |
| 李霖 | 2022/6/7 | 1. 更新席位信息协议，见表8。 | | | V1.0.1.11 |
| 李霖 | 2022/8/13 | 1) 添加组合合约推送功能，见章节5.3.7；  2) 添加资金查询功能，见章节5.2.17和5.3.39；  3） 添加持仓查询功能，见章节5.2.18和5.3.38；  4） 添加初始化查询信息设置功能，用户可配置API初始化期间查询的内容，见章节5.2.9，同时修改初始化函数Init()接口，见章节5.2.10。 | | | V1.0.1.12 |
| 李霖 | 2022/9/7 | 修复Windows下API初始化过程中偶发断开连接的问题。 | | | V1.0.1.13 |

目录

[1. 简介 1](#_Toc111291439)

[2. 类库文件 1](#_Toc111291440)

[3. 接口模式 1](#_Toc111291441)

[4. 使用流程 1](#_Toc111291442)

[4.1 基本流程 1](#_Toc111291443)

[4.2 登录认证 3](#_Toc111291444)

[5. 接口说明 3](#_Toc111291445)

[5.1接口创建和释放 3](#_Toc111291446)

[5.1.1 CreateDstarTradeApi 3](#_Toc111291447)

[5.1.2 FreeDstarTradeApi 3](#_Toc111291448)

[5.2 IDstarTradeApi接口 3](#_Toc111291449)

[5.2.1 RegisterSpi 3](#_Toc111291450)

[5.2.2 RegisterFrontAddress 4](#_Toc111291451)

[5.2.3 SetApiLogPath 4](#_Toc111291452)

[5.2.4 SetLoginInfo 4](#_Toc111291453)

[5.2.5 SetCpuId 4](#_Toc111291454)

[5.2.6 SetSubscribeStartId 4](#_Toc111291455)

[5.2.7 GetSystemInfo 5](#_Toc111291456)

[5.2.8 SetSubmitInfo 5](#_Toc111291457)

[5.2.9 SetInitQryInfo 5](#_Toc111291458)

[5.2.10 Init 5](#_Toc111291459)

[5.2.11 ReqPwdMod 5](#_Toc111291460)

[5.2.12 ReqLastClientReqId 6](#_Toc111291461)

[5.2.13 ReqOrderInsert 6](#_Toc111291462)

[5.2.14 ReqOfferInsert 6](#_Toc111291463)

[5.2.15 ReqOrderDelete 6](#_Toc111291464)

[5.2.16 ReqCmbOrderInsert 6](#_Toc111291465)

[5.2.17 ReqQryFund 6](#_Toc111291466)

[5.2.18 ReqQryPosition 7](#_Toc111291467)

[5.2.19 GetApiVersion 7](#_Toc111291468)

[5.3 IDstarTradeSpi接口 7](#_Toc111291469)

[5.3.1 OnFrontDisconnected 7](#_Toc111291470)

[5.3.2 OnRspError 7](#_Toc111291471)

[5.3.3 OnRspUserLogin 7](#_Toc111291472)

[5.3.4 OnRspPwdMod 8](#_Toc111291473)

[5.3.5 OnRspSubmitInfo 8](#_Toc111291474)

[5.3.6 OnRspContract 8](#_Toc111291475)

[5.3.7 OnRspCmbContract 8](#_Toc111291476)

[5.3.8 OnRspSeat 8](#_Toc111291477)

[5.3.9 OnRspTrdFeeParam 9](#_Toc111291478)

[5.3.10 OnRspTrdMarParam 9](#_Toc111291479)

[5.3.11 OnRspTradeRight 9](#_Toc111291480)

[5.3.12 OnRspAccountCommList 9](#_Toc111291481)

[5.3.13 OnRspTrdExchangeState 10](#_Toc111291482)

[5.3.14 OnRspFund 10](#_Toc111291483)

[5.3.15 OnRspPrePosition 10](#_Toc111291484)

[5.3.16 OnRspPosition 10](#_Toc111291485)

[5.3.17 OnRspOrder 11](#_Toc111291486)

[5.3.18 OnRspOffer 11](#_Toc111291487)

[5.3.19 OnRspMatch 11](#_Toc111291488)

[5.3.20 OnRspCashInOut 11](#_Toc111291489)

[5.3.21 OnApiReady 12](#_Toc111291490)

[5.3.22 OnRspUdpAuth 12](#_Toc111291491)

[5.3.23 OnRspOrderInsert 12](#_Toc111291492)

[5.3.24 OnRspOfferInsert 12](#_Toc111291493)

[5.3.25 OnRspOrderDelete 12](#_Toc111291494)

[5.3.26 OnRspLastReqId 13](#_Toc111291495)

[5.3.27 OnRtnPwdMod 13](#_Toc111291496)

[5.3.28 OnRtnOrder 13](#_Toc111291497)

[5.3.29 OnRtnMatch 13](#_Toc111291498)

[5.3.30 OnRtnCashInOut 14](#_Toc111291499)

[5.3.31 OnRtnOffer 14](#_Toc111291500)

[5.3.32 OnRtnEnquiry 14](#_Toc111291501)

[5.3.33 OnRtnTrdExchangeState 14](#_Toc111291502)

[5.3.34 OnRtnPosiProfit 14](#_Toc111291503)

[5.3.35 OnRtnSeat 15](#_Toc111291504)

[5.3.36 OnRtnTradeRight 15](#_Toc111291505)

[5.3.37 OnRtnTradeRightDel 15](#_Toc111291506)

[5.3.38 OnRspQryPosition 15](#_Toc111291507)

[5.3.39 OnRspQryFund 15](#_Toc111291508)

[6. 资金计算 16](#_Toc111291509)

[6.1 从头计算 16](#_Toc111291510)

[6.2 从最新计算 16](#_Toc111291511)

[7. 持仓计算 16](#_Toc111291512)

[8. 看穿式监管 17](#_Toc111291513)

[9. 报撤单 17](#_Toc111291514)

[9.1批量报撤单 17](#_Toc111291515)

[9.2撤单通知 17](#_Toc111291516)

[10. UDP协议使用样例 17](#_Toc111291517)

[10.1 UDP认证样例 17](#_Toc111291518)

[10.2 UDP报撤单样例 18](#_Toc111291519)

[10.3 询价单样例 21](#_Toc111291520)

[10.4 报价单样例 21](#_Toc111291521)

[11. 协议变更 23](#_Toc111291522)

# 简介

本文档描述了启明星V10交易系统用户接口规范，为使用启明星V10交易系统的投资者提供使用参考。

启明星V10交易系统用户接口(Api)是一个基于C++的类库，提供Linux系统及Windows系统版本动态库。

# 类库文件

表 1 Api包文件描述

|  |  |
| --- | --- |
| 文件名 | 描述 |
| DstarTradeApi.h | 交易接口头文件 |
| DstarTradeApiError.h | 错误码头文件 |
| DstarTradeApiDataType.h | 数据类型头文件 |
| DstarTradeApiStruct.h | 数据结构头文件 |
| libdstartradeapi.so | Linux下Api动态库文件 |
| libdstartradeapi.lib | Windows下Api导入库文件 |
| libdstartradeapi.dll | Windows下Api动态链接库二进制文件 |

# 接口模式

Api由调用接口类(IDstarTradeApi)和回调接口类(IDstarTradeSpi)两部分组成。IDstarTradeApi提供了用户可以使用的接口函数，IDstarTradeSpi提供了用户回调接口。用户需要从IDstarTradeSpi派生子类重载回调函数处理后台服务响应。

# 使用流程

## 4.1 基本流程

用户与交易系统数据通信包含两部分，一是通过Api接收数据, 二是通过UDP进行报撤单操作。

Api接收的数据包括登录应答、席位信息、合约信息、持仓快照、资金快照、委托查询、成交查询等查询数据和认证应答、报撤单应答、委托通知、成交通知、出入金通知等通知数据。

用户使用Api和UDP报撤单基本流程如下图所示。

初始化过程中数据回调

OnRspUserLogin:用户登录响应

OnRspSubmitInfo:提交信息响应

OnRspContract:合约信息响应

OnRspCmbContract:组合合约信息响应

OnRspSeat:席位信息响应

OnRspTrdFeeParam:手续费参数响应

OnRspTrdMarParam:保证金参数响应

OnRspTradeRight:交易权限响应

OnRspAccountCommList:客户品种白名单响应

OnRspTrdExchangeState:市场状态信息响应

OnRspFund:资金快照响应

OnRspPrePosition:昨持仓快照响应

OnRspPosition:实时持仓快照响应

OnRspOrder:委托查询响应

OnRspOffer:报价响应

OnRspMatch:成交查询响应

OnRspCashInOut:出入金响应

RegisterSpi:注册回调对象

RegisterFrontAddress:注册服务器地址

SetApiLogPath:设置工作目录

SetLoginInfo:设置用户信息

SetCpuId:Api线程绑核

SetSubscribeStartId:设置订阅位置

SetSubmitInfo:设置采集信息

Init

Api初始化

Api初始化数据回调

用户报单

用户UDP报文

OnRspUdpAuth

UDP认证应答

UDP认证

用户UDP报文

OnApiReady

Api初始化完成

CreateDstarTradeApi

创建Api实例

Api设置

**(必须在Init之前调用)**

## 4.2 登录认证

用户设置登录信息，登录过程由Api来完成。

登录应答回调返回UDP认证码，UDP认证码是动态的，每个认证码对应一次登录。Api初始化完成后，用户使用裸协议进行UDP认证。

交易服务会绑定UDP认证码、UDP认证请求的ip、端口。后续的报撤单交易会对UDP认证码、报撤单ip、端口进行订单来源检测。当Api连接断开，UDP认证码失效，用户需要重新进行认证。

认证时需要指定请求号模式，请求号模式包括不检测、增大、强制自增。不检测模式时，请求号无要求；增大模式时，请求号要大于前一包数据的请求号，否则报撤单失败；强制自增模式时，请求号必须连续自增，否则报撤单失败。

# 接口说明

## 5.1接口创建和释放

### 5.1.1 CreateDstarTradeApi

生成一个Api实例。

函数原型：

|  |
| --- |
| IDstarTradeApi \*CreateDstarTradeApi(); |

说明：

**Api不是线程安全的，不能在多个线程中使用同一个Api实例。**

**不要在Api回调接口中进行过多的业务处理，以免影响以免影响实时数据的接收速度。**

### 5.1.2 FreeDstarTradeApi

释放一个Api实例。

函数原型：

|  |
| --- |
| void FreeDstarTradeApi(IDstarTradeApi \*pApiObj); |

说明：

pApiObj,Api实例指针。

## 5.2 IDstarTradeApi接口

### RegisterSpi

注册回调对象实例。

函数原型：

|  |
| --- |
| virtual void RegisterSpi(IDstarTradeSpi \*pSpi) = 0; |

### RegisterFrontAddress

注册接口地址。

函数原型：

|  |
| --- |
| virtual void RegisterFrontAddress(DstarApiIpType pIp, DstarApiPortType nPort) = 0; |

说明：

pIp,服务器地址。

nPort,服务器端口。

### SetApiLogPath

设置Api保存交易数据文件目录和工作日志文件目录。

函数原型：

|  |
| --- |
| virtual void SetApiLogPath(DstarApiPathType pPath) = 0; |

说明：

pPath,日志路径，路径必须可用。

### SetLoginInfo

设置用户信息

函数原型：

|  |
| --- |
| virtual void SetLoginInfo(DstarApiReqLoginField \*pLoginInfo) = 0; |

说明：

pLoginInfo,登录信息，包括用户名、密码、AppId、LicenseNo、协议版本号。

### SetCpuId

设置Api线程绑定的CPU Id。

函数原型：

|  |
| --- |
| virtual void SetCpuId(DstarApiCpuIdType nRecvNoticeDataCpuId, DstarApiCpuIdType nLogCpuId) = 0; |

说明：

nRecvNoticeDataCpuId,接收通知数据线程Cpu Id,必须绑定。

nLogCpuId,日志线程Cpu Id,可选。-1:不绑定,同时不记录交易数据日志。

### SetSubscribeStartId

设置通知流订阅起始位置。

函数原型：

|  |
| --- |
| virtual void SetSubscribeStartId(DstarApiNoticeSubIdType nStartId) = 0; |

说明：

nStartId,通知流订阅位置，-1:从最新开始订阅 0:从头订阅，>0:从指定位置订阅。

### GetSystemInfo

采集系统信息。

函数原型：

|  |
| --- |
| virtual int GetSystemInfo(char\* pSystemInfo, int \*nLen, unsigned int \*nAuthKeyVersion) = 0; |

说明：

pSystemInfo,存放采集信息,由用户申请空间,长度不小于500字节。

nLen,返回采集信息的长度。

nAuthKeyVersion,返回密钥版本。

### SetSubmitInfo

设置提交信息。

函数原型：

|  |
| --- |
| virtual void SetSubmitInfo(DstarApiSubmitInfoField \*pSubmitInfo) = 0; |

说明：

pSubmitInfo,提交信息。用户需要填写GetSystemInfo返回的采集信息和密钥版本，用户登录相关信息（登录Ip和Port、软件授权号和AppId等信息）。

### SetInitQryInfo

设置初始化查询信息。

函数原型：

|  |
| --- |
| virtual void SetInitQryInfo(DstarApiInitQryInfoField \*pInitQryInfo) = 0; |

说明：

pInitQryInfo,查询信息。填写用户初始化需要查询的信息。

### Init

Api初始化。

函数原型：

|  |
| --- |
| virtual int Init() = 0; |

说明:

nInitFlag,Api初始化标志。默认查询所有信息，用户可以通过SetInitQryInfo函数设置需要查询的数据项，提高初始化速度。

### ReqPwdMod

密码修改请求。

函数原型：

|  |
| --- |
| virtual int ReqPwdMod(const DstarApiReqPwdModField \*pReqPwdModField) = 0; |

说明：

用户可填入新密码和旧密码进行账户密码修改操作。

### ReqLastClientReqId

查询最新客户请求号。

函数原型：

|  |
| --- |
| virtual int ReqLastClientReqId() = 0; |

说明：

用户可以通过查询最新请求号检测报撤单是否有丢包情况。查询间隔不小于5s。

### ReqOrderInsert

输入报单请求。

函数原型：

|  |
| --- |
| virtual int ReqOrderInsert(const DstarApiReqOrderInsertField \*pOrder) = 0; |

### ReqOfferInsert

输入报价请求。

函数原型：

|  |
| --- |
| virtual int ReqOfferInsert(const DstarApiReqOfferInsertField \*pOffer) = 0; |

### ReqOrderDelete

撤单请求。

函数原型：

|  |
| --- |
| virtual int ReqOrderDelete(const DstarApiReqOrderDeleteField \*pOrder) = 0; |

### ReqCmbOrderInsert

输入组合报单请求。

函数原型：

|  |
| --- |
| virtual int ReqCmbOrderInsert(const DstarApiReqCmbOrderInsertField \*pCmbOrder) = 0; |

### ReqQryFund

资金查询请求。

函数原型：

|  |
| --- |
| virtual int ReqQryFund() = 0; |

### ReqQryPosition

持仓查询请求。

函数原型：

|  |
| --- |
| virtual int ReqQryPosition() = 0; |

### GetApiVersion

获取Api版本信息。

函数原型：

|  |
| --- |
| virtual const char \*GetApiVersion() = 0; |

## 5.3 IDstarTradeSpi接口

### OnFrontDisconnected

客户端与接口通信连接断开。

函数原型：

|  |
| --- |
| virtual void OnFrontDisconnected() = 0; |

说明：

Api与交易服务连接断开后回调。**断开后当前的UDP认证同时失效。**

**断线后Api不会自动重连，需要用户重新调用Init接口重新初始化Api。收到OnApiReady回调后重新进行UDP认证。**

**或者释放Api实例对象，创建新的Api实例对象重新开始处理。**

### OnRspError

错误应答。

函数原型：

|  |
| --- |
| virtual void OnRspError(DstarApiErrorCodeType nErrorCode) = 0; |

说明：

Api运行过程发生错误，回调通知用户错误码。

错误码详情查看< DstarTradeApiDataType.h>头文件

### OnRspUserLogin

登录请求响应。

函数原型：

|  |
| --- |
| virtual void OnRspUserLogin(const DstarApiRspLoginField \*pRspUserLogin) = 0; |

说明：

Api初始化过程登录请求的登录请求响应。结构体中包含有UDP认证的认证码，认证码用于UDP认证。UDP认证码绑定连接，认证之后报撤单不能切换连接。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspPwdMod

密码修改响应。

函数原型：

|  |
| --- |
| virtual void OnRspPwdMod(const DstarApiRspPwdModField \*pRspPwdModField) = 0; |

说明：

根据反馈的错误码查看密码是否修改成功。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspSubmitInfo

提交信息响应。

函数原型：

|  |
| --- |
| virtual void OnRspSubmitInfo(const DstarApiRspSubmitInfoField \*pRspSubmitInfo) = 0; |

说明：

提交信息响应。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspContract

合约信息响应。

函数原型：

|  |
| --- |
| virtual void OnRspContract(const DstarApiContractField \*pContract) = 0; |

说明：

合约信息响应，同一个交易日内数据相同。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspCmbContract

组合合约信息响应。

函数原型：

|  |
| --- |
| virtual void OnRspCmbContract(const DstarApiCmbContractField \*pCmbContract) = 0; |

说明：

组合合约信息响应，同一个交易日内数据相同。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspSeat

席位信息响应。

函数原型：

|  |
| --- |
| virtual void OnRspSeat(const DstarApiSeatField\* pSeat) = 0; |

说明：

席位信息响应，同一个交易日内数据相同。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspTrdFeeParam

手续费参数响应。

函数原型：

|  |
| --- |
| virtual void OnRspTrdFeeParam(const DstarApiTrdFeeParamField\* pFeeParam) = 0; |

说明：

手续费参数响应，同一个交易日内数据相同。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspTrdMarParam

保证金参数响应。

函数原型：

|  |
| --- |
| virtual void OnRspTrdMarParam(const DstarApiTrdMarParamField\* pMarParam) = 0; |

说明：

保证金参数响应，同一个交易日内数据相同。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspTradeRight

交易权限响应。

函数原型：

|  |
| --- |
| virtual void OnRspTradeRight(const DstarApiTradeRightField\* pTradeRight) = 0; |

说明：

交易权限响应。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspAccountCommList

客户品种白名单响应。

函数原型：

|  |
| --- |
| virtual void OnRspAccountCommList(const DstarApiAccountCommListField\* pAccountCommList) = 0; |

说明：

客户品种白名单响应，同一个交易日内数据相同。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspTrdExchangeState

市场状态信息响应。

函数原型：

|  |
| --- |
| virtual void OnRspTrdExchangeState(const DstarApiTrdExchangeStateField \*pTrdExchangeState) = 0; |

说明：

市场状态信息响应。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspFund

资金快照响应。

函数原型：

|  |
| --- |
| virtual void OnRspFund(const DstarApiFundField \*pFund) = 0; |

说明：

资金快照响应。快照数据包含昨资金和今资金数据，是服务器中的实时数据。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspPrePosition

昨持仓快照响应。

函数原型：

|  |
| --- |
| virtual void OnRspPrePosition(const DstarApiPrePositionField \*pPrePosition) = 0; |

说明：

昨持仓快照响应。快照数据为持仓合约合计数据，是用户初始昨持仓，当前交易日保持不变。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspPosition

实时持仓快照响应。

函数原型：

|  |
| --- |
| virtual void OnRspPosition(const DstarApiPositionField \*pPosition) = 0; |

说明：

实时持仓快照响应。快照数据为持仓合约合计数据，包含昨持仓和今持仓数据，是服务器中的实时数据，昨持仓是用户最新的昨持仓数据。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspOrder

委托响应。

函数原型：

|  |
| --- |
| virtual void OnRspOrder(const DstarApiOrderField \*pOrder) = 0; |

说明：

委托响应。Api初始化过程中会查询客户委托，返回客户当前所有委托数据。委托数据中包含冻结保证金数据，该数据是账户实时资金数据。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspOffer

报价响应。

函数原型：

|  |
| --- |
| virtual void OnRspOffer(const DstarApiOfferField \*pOffer) = 0; |

说明：

报价响应。Api初始化过程中会查询客户报价，返回客户当前所有报价数据。报价数据中包含冻结保证金数据，该数据是账户实时资金数据。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspMatch

成交响应。

函数原型：

|  |
| --- |
| virtual void OnRspMatch(const DstarApiMatchField \*pMatch) = 0; |

说明：

成交响应。Api初始化过程中会查询客户成交，返回客户当前所有成交数据。成交数据中包含手续费、保证金、冻结保证金、权利金(或平仓盈亏)数据，这些数据是账户实时资金数据。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspCashInOut

出入金响应。

函数原型：

|  |
| --- |
| virtual void OnRspCashInOut(const DstarApiCashInOutField \*pCashInOut) = 0; |

说明：

出入金响应。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnApiReady

Api准备就绪。

函数原型：

|  |
| --- |
| virtual void OnApiReady(const DstarApiSerialIdType nSerialId) = 0; |

说明：

用户收到此回调说明Api完成初始化工作，用户可在此回调进行UDP认证。

nSerialId,快照数据(资金、持仓)对应的数据流号。

### OnRspUdpAuth

UDP认证响应。

函数原型：

|  |
| --- |
| virtual void OnRspUdpAuth(const DstarApiRspUdpAuthField \*pRspUdpAuth) = 0; |

说明：

UDP认证应答。认证成功后才可以报撤单。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspOrderInsert

报单应答。

函数原型：

|  |
| --- |
| virtual void OnRspOrderInsert(const DstarApiRspOrderInsertField \*pOrderInsert) = 0; |

说明：

报单应答。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspOfferInsert

报价应答。

函数原型：

|  |
| --- |
| virtual void OnRspOfferInsert(const DstarApiRspOfferInsertField \*pOfferInsert) = 0; |

说明：

报价应答。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspOrderDelete

撤单应答。

函数原型：

|  |
| --- |
| virtual void OnRspOrderDelete(const DstarApiRspOrderDeleteField \*pOrderDelete) = 0; |

说明：

撤单应答。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspLastReqId

最新请求号应答。

函数原型：

|  |
| --- |
| virtual void OnRspLastReqId(const DstaApiRspLastReqIdField \*pLastReqId) = 0; |

说明：

最新请求号应答。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRtnPwdMod

密码修改通知。

函数原型：

|  |
| --- |
| virtual void OnRtnPwdMod(const DstarApiPwdModField \*pPwdModField) = 0; |

说明：

收到此通知后API会主动断开连接。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRtnOrder

委托通知。

函数原型：

|  |
| --- |
| virtual void OnRtnOrder(const DstarApiOrderField \*pOrder) = 0; |

说明：

委托通知。委托通知数据中包含冻结保证金数据，该数据是账户实时资金数据。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRtnMatch

成交通知。

函数原型：

|  |
| --- |
| virtual void OnRtnMatch(const DstarApiMatchField \*pMatch) = 0; |

说明：

成交通知。成交通知数据中包含手续费、保证金、冻结保证金、权利金(或平仓盈亏)数据，这些数据是账户实时资金数据。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRtnCashInOut

出入金通知。

函数原型：

|  |
| --- |
| virtual void OnRtnCashInOut(const DstarApiCashInOutField \*pCashInOut) = 0; |

说明：

出入金通知。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRtnOffer

报价通知。

函数原型：

|  |
| --- |
| virtual void OnRtnOffer(const DstarApiOfferField \*pOffer) = 0; |

说明：

报价通知。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRtnEnquiry

询价通知。

函数原型：

|  |
| --- |
| virtual void OnRtnEnquiry(const DstarApiEnquiryField \*pEnquiry) = 0; |

说明：

询价通知。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRtnTrdExchangeState

市场状态信息通知。

函数原型：

|  |
| --- |
| virtual void OnRtnTrdExchangeState(const DstarApiTrdExchangeStateField \*pTrdExchangeState) = 0; |

说明：

市场状态信息通知。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRtnPosiProfit

浮盈通知。

函数原型：

|  |
| --- |
| virtual void OnRtnPosiProfit(const DstarApiPosiProfitField \*pPosiProfit) = 0; |

说明：

浮盈通知。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRtnSeat

席位信息通知。

函数原型：

|  |
| --- |
| virtual void OnRtnSeat(const DstarApiSeatField\* pSeat) = 0; |

说明：

席位信息通知。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRtnTradeRight

交易权限通知。

函数原型：

|  |
| --- |
| virtual void OnRtnTradeRight(const DstarApiTradeRightField\* pTradeRight) = 0; |

说明：

交易权限通知，当添加或者修改某个品种交易权限时，会推送此通知。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRtnTradeRightDel

交易权限删除通知。

函数原型：

|  |
| --- |
| virtual void OnRtnTradeRightDel(const DstarApiTradeRightDelField\* pTradeRightDel) = 0; |

说明：

交易权限通知，当删除某个品种交易权限时，会推送此通知。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspQryPosition

持仓查询响应。

函数原型：

|  |
| --- |
| virtual void OnRspQryPosition(const DstarApiPositionField \*pPosition, bool bLast) = 0; |

说明：

持仓查询响应，当调用持仓查询接口时，会推送此通知。

结构体详情查看<DstarTradeApiStruct.h>头文件。

### OnRspQryFund

资金查询响应。

函数原型：

|  |
| --- |
| virtual void OnRspQryFund(const DstarApiFundField \*pFund) = 0; |

说明：

资金查询通知，当调用资金查询接口时，会推送此通知。

结构体详情查看<DstarTradeApiStruct.h>头文件。

# 资金计算

资金计算的数据来源包括资金快照、委托通知、报价通知、成交通知、出入金通知数据。

委托通知、报价通知、成交通知中的资金数据是账户实时资金数据。例如委托通知中的冻结保证金，是该笔委托更新后账户总的冻结保证金，不是该笔委托的冻结保证金。

用户计算资金有两种方式，一种是使用资金快照中的昨资金(昨权益值)，从头计算；一种是使用资金快照中的今资金数据，从最新计算。

计算过程中，使用委托通知中的冻结保证金数据；使用成交通知中的手续费、保证金、冻结保证金、权利金(或平仓盈亏)数据；使用出入金的出入金类型和出入金数金额数据。

## 从头计算

用户使用资金快照中的昨权益作为初始数据，收到委托通知、报价通知、成交通知、出入金通知时进行计算更新。从头计算需要从头订阅通知数据。

## 从最新计算

Api完成初始化回调的OnApiReady中返回快照数据对应的流号。这个流号是已参与资金快照计算的最新数据流号。

用户使用资金快照作为初始数据，收到委托通知、报价通知、成交通知、出入金通知时计算更新。计算更新时，需要先判断通知数据上的流号是否大于快照流号。如果大于快照流号，说明是新的数据，进行计算；小于等于快照流号，说明是已计算过的数据，不再计算。

# 持仓计算

持仓的计算方式和资金计算一样，可以从头计算或从最新计算。

从头计算时，使用昨持仓快照中的昨持仓数据作为初始数据，收到成交通知时计算更新。

从最新计算时，使用持仓快照中最新的昨持仓和今持仓作为初始数据，收到成交通知时计算更新。计算更新时，需要先判断成交通知数据上的流号是否大于持仓快照流号。如果大于持仓快照流号，说明是新的数据，进行计算；小于等于持仓快照流号，说明是已计算过的数据，不再计算。

# 看穿式监管

用户设置登录信息时，需要AppId和LicenseNo。登录会认证AppId和LicenseNo,认证不通过时登录失败。如果是直连模式，AppId为客户AppId；如果是中继模式，AppId为中继AppId。

用户需要在Api初始化之前，设置上报信息。

用户通过GetSystemInfo接口获取系统信息和密钥版本，通过SetSubmitInfo设置上报信息。上报信息包括系统信息、密钥版本、账号、授权类型、客户登录Ip、客户登录端口、客户登录时间、客户AppId、软件授权号。其中，客户登录Ip、端口和登录时间，如果是直连模式，不需要填写；如果是中继模式，需要填写。

Api在登录成功之后，发送设置的上报信息，登录失败不发送。断线重连时，需要重新采集系统信息，重新设置上报信息。

# 报撤单

## 9.1批量报撤单

对于支持批量报单的系统，可进行批量报单、批量撤单、批量报价。对于不支持批量报单的系统如果进行批量报撤单，只会通过第一笔订单。

## 9.2撤单通知

用户撤单时，会收到被撤单子的委托通知或报价通知。

当撤单成功时，通知的委托状态为已撤单。

当撤单失败时，通知的委托状态为单子的当前状态，报单引用为撤单操作的报单引用，错误码为撤单失败的错误码。

# 10. UDP协议使用样例

UDP协议包含协议头和协议主体两部分内容，在Api包中**协议头和协议主体为独立的结构体**，**为1字节对齐方式**。其中协议头包含协议号和数据长度，数据长度为协议主体的长度，并不包含协议头；协议主体为用户所需填报的字段。**单个报文长度最大为1024个字节。UDP协议支持单个报文内填充多个不同类型协议消息的功能，例如同一包报文可以同时包含报单和撤单(仅针对支持批量报单的系统)。**

## 10.1 UDP认证样例

UDP认证由用户直接发送报文到交易服务。

下面是发送认证报文的例子：

|  |
| --- |
| char sendbuf[256] = {0};  DstarApiHead \*head = (DstarApiHead \*)sendbuf;  head->ProtocolCode = CMD\_API\_Req\_UdpAuth;  head->DataLen = **sizeof(DstarApiReqUdpAuthField);**  DstarApiReqUdpAuthField \*req = (DstarApiReqUdpAuthField \*)&sendbuf[sizeof(DstarApiHead)];  req->AccountIndex = m\_LoginInfo.AccountIndex; //登录应答信息中的用户索引  req->UdpAuthCode = m\_LoginInfo.UdpAuthCode; //登录应答信息中的认证码  req->ReqIdMode = DSTAR\_API\_REQIDMODE\_FORCE; //**0:不检测 1:增大 2:强制连续自增**  Send(sendbuf, sizeof(DstarApiReqUdpAuthField) + sizeof(DstarApiHead)); //UDP发送 |

## 10.2 UDP报撤单样例

用户报撤单操作由用户直接发送报文到交易服务。

下面是发送一个报单请求的例子：

|  |
| --- |
| char sendbuf[256] = {0};  DstarApiHead \*head = (DstarApiHead \*)sendbuf;  head->ProtocolCode = CMD\_API\_Req\_OrderInsert;  head->DataLen = sizeof(DstarApiReqOrderInsertField);  DstarApiReqOrderInsertField \*req = (DstarApiReqOrderInsertField \*)&sendbuf[sizeof(DstarApiHead)];  req->Direct = DSTAR\_API\_DIRECT\_BUY;  req->Offset = DSTAR\_API\_OFFSET\_OPEN;  req->Hedge = DSTAR\_API\_HEDGE\_SPECULATE;  req->OrderType = DSTAR\_API\_ORDERTYPE\_LIMIT;  req->ValidType = DSTAR\_API\_VALID\_FOK;  req->SeatIndex = 1; **//席位信息中的席位索引,填0不指定席位，由交易服务轮询席位**  req->AccountIndex = m\_LoginInfo.AccountIndex; //登录应答信息中的用户索引  req->UdpAuthCode = m\_LoginInfo.UdpAuthCode; //登录应答信息中的认证码  req->ClientReqId = ClientReqId++; **//根据认证时的请求号模式填写**  req->ContractIndex = contractindex; //合约信息中的合约索引  strncpy(req->ContractNo, ContractNo, sizeof(DstarApiContractNoType) - 1);  req->OrderQty = nQty;  req->OrderPrice = price;  req->MinQty = 0;  req->Reference = 1; //要求 >=0  Send(sendbuf, sizeof(DstarApiReqOrderInsertField) + sizeof(DstarApiHead)); //UDP发送 |

下面是发送一个组合报单请求的例子：

|  |
| --- |
| char sendbuf[256] = {0};  DstarApiHead \*head = (DstarApiHead \*)sendbuf;  head->ProtocolCode = CMD\_API\_Req\_CmbOrderInsert;  head->DataLen = sizeof(DstarApiReqCmbOrderInsertField);  DstarApiReqCmbOrderInsertField\*req = (DstarApiReqCmbOrderInsertField\*)&sendbuf[sizeof(DstarApiHead)];  req->Direct = DSTAR\_API\_DIRECT\_BUY;  req->Offset = DSTAR\_API\_OFFSET\_OPEN;  req->Hedge = DSTAR\_API\_HEDGE\_SPECULATE;  req->OrderType = DSTAR\_API\_ORDERTYPE\_LIMIT;  req->ValidType = DSTAR\_API\_VALID\_IOC;  req->SeatIndex = 1; **//席位信息中的席位索引,填0不指定席位，由交易服务轮询席位**  req->AccountIndex = m\_LoginInfo.AccountIndex; //登录应答信息中的用户索引  req->UdpAuthCode = m\_LoginInfo.UdpAuthCode; //登录应答信息中的认证码  req->ClientReqId = ClientReqId++; **//根据认证时的请求号模式填写**  req->ContractIndex1 = contractindex1; //合约信息中的合约索引1  strncpy(req->ContractNo1, ContractNo1, sizeof(DstarApiContractNoType) - 1);  req->ContractIndex2 = contractindex2; //合约信息中的合约索引2  strncpy(req->ContractNo2, ContractNo2, sizeof(DstarApiContractNoType) - 1);  req->OrderQty = nQty;  req->OrderPrice = price; // **期货跨期、跨品种套利填写价格 = 合约1价格 – 合约2价格**  **// 期权（宽）跨式组合填写价格 = 合约1价格 + 合约2价格**  req->MinQty = 0;  req->Reference = 1; //要求 >=0  Send(sendbuf, sizeof(DstarApiReqCmbOrderInsertField) + sizeof(DstarApiHead)); //UDP发送 |

下面是发送批量报单请求的例子：

|  |
| --- |
| char sendbuf[1024] = {0};  int num = 6;  DstarApiHead \*head = NULL;  DstarApiReqOrderInsertField \*req = NULL;  int pos = 0;  for (int i = 0; i < num; i++)  {  head = (DstarApiHead\*)& sendbuf[pos];  head->ProtocolCode = CMD\_API\_Req\_OrderInsert;  head->DataLen = sizeof(DstarApiReqOrderInsertField);  req=(DstarApiReqOrderInsertField \*) &sendbuf[pos + sizeof(DstarApiHead)];  req->Direct = DSTAR\_API\_DIRECT\_BUY;  req->Offset = DSTAR\_API\_OFFSET\_OPEN;  req->Hedge = DSTAR\_API\_HEDGE\_SPECULATE;  req->OrderType = DSTAR\_API\_ORDERTYPE\_LIMIT;  req->ValidType = DSTAR\_API\_VALID\_FOK;  req->SeatIndex = 0; **//席位信息中的席位索引,填0不指定席位，由交易服务轮询席位**  req->AccountIndex = m\_LoginInfo.AccountIndex; //登录应答信息中的用户索引  req->UdpAuthCode = m\_LoginInfo.UdpAuthCode; //登录应答信息中的认证码  req->ClientReqId = ClientReqId++;  req->ContractIndex = contractindex; //合约信息中的合约索引  strncpy(req->ContractNo, ContractNo, sizeof(DstarApiContractNoType) - 1);  req->OrderQty = nQty;  req->OrderPrice = price;  req->MinQty = 0;  req->Reference = 1; //要求 >=0  pos += sizeof(DstarApiReqOrderInsertField) + sizeof(DstarApiHead);  }  Send(sendbuf, pos); //UDP发送 |

下面是发送撤单请求的例子：

|  |
| --- |
| char sendbuf[256] = {0};  DstarApiHead \*head = (DstarApiHead \*)sendbuf;  head->ProtocolCode = CMD\_API\_Req\_OrderDelete;  head->DataLen = sizeof(DstarApiReqOrderDeleteField);  DstarApiReqOrderDeleteField\* req =  (DstarApiReqOrderDeleteField \*)&sendbuf[sizeof(DstarApiHead)];  req->AccountIndex = m\_LoginInfo.AccountIndex;  req->ClientReqId = ClientReqId++;  req->UdpAuthCode = m\_LoginInfo.UdpAuthCode;  req->SeatIndex = 0; **//0从报单席位撤单,非0从指定席位撤单**  req->Reference = 0; //要求 >=0  req->OrderId = m\_OrderNotice.OrderId; **// 对委托号相对应系统号的单子进行撤单(此字段为必填项)**  strncpy(req->SystemNo, SystemNo, sizeof(DstarApiSystemNoType) - 1); **// 直接对对应系统号的单子进行撤单(此字段为非必填项)**  Send(sendbuf, sizeof(starApiReqOrderDeleteField) + sizeof(DstarApiHead)); //UDP发送 |

下面是发送批量撤单请求的例子：

|  |
| --- |
| char sendbuf[1024] = {0};  int num = 6;  DstarApiHead \*head = NULL;  DstarApiReqOrderDeleteField \*req = NULL;  int pos = 0;  for (int i = 0; i < num; i++)  {  head = (DstarApiHead \*)&sendbuf[pos];  head->ProtocolCode = CMD\_API\_Req\_OrderDelete;  head->DataLen = sizeof(DstarApiReqOrderDeleteField);  req=(DstarApiReqOrderDeleteField \*) &sendbuf[pos + sizeof(DstarApiHead)];  req->AccountIndex = m\_LoginInfo.AccountIndex;  req->ClientReqId = ClientReqId++;  req->UdpAuthCode = m\_LoginInfo.UdpAuthCode;  req->SeatIndex = 0; **//0从报单席位撤单,非0从指定席位撤单**  req->Reference = 0; //要求 >=0  req->OrderId = m\_OrderNotice.OrderId;  strncpy(req->SystemNo, SystemNo, sizeof(DstarApiSystemNoType) - 1);  }  Send(sendbuf, pos); //UDP发送 |

## 10.3 询价单样例

下面是发送一个询价单请求的例子：

|  |
| --- |
| char sendbuf[256] = {0};  DstarApiHead \*head = (DstarApiHead \*)sendbuf;  head->ProtocolCode = CMD\_API\_Req\_OrderInsert;  head->DataLen = sizeof(DstarApiReqOrderInsertField);  DstarApiReqOrderInsertField \*req =  (DstarApiReqOrderInsertField \*)&sendbuf[sizeof(DstarApiHead)];  req->OrderType = **DSTAR\_API\_ORDERTYPE\_ENQUIRY**;  req->SeatIndex = 1; **//席位信息中的席位索引,填0不指定席位，由交易服务轮询席位**  req->AccountIndex = m\_LoginInfo.AccountIndex; //登录应答信息中的用户索引  req->UdpAuthCode = m\_LoginInfo.UdpAuthCode; //登录应答信息中的认证码  req->ClientReqId = ClientReqId++;  req->ContractIndex = contractindex; //合约信息中的合约索引  strncpy(req->ContractNo, ContractNo, sizeof(DstarApiContractNoType) - 1);  req->Reference = 1; //要求 >=0  Send(sendbuf, sizeof(DstarApiReqOrderInsertField) + sizeof(DstarApiHead)); //UDP发送 |

## 10.4 报价单样例

下面是发送一个报价单请求的例子：

|  |
| --- |
| char sendbuf[256] = {0};  DstarApiHead \*head = (DstarApiHead \*)sendbuf;  head->ProtocolCode = **CMD\_API\_Req\_OfferInsert**;  head->DataLen = sizeof(DstarApiReqOfferInsertField);  DstarApiReqOfferInsertField \*req =  (DstarApiReqOfferInsertField \*)&sendbuf[sizeof(DstarApiHead)];  req->BuyOffset = DSTAR\_API\_OFFSET\_OPEN;  req->SellOffset = DSTAR\_API\_OFFSET\_OPEN;  req->OrderQty = qty;  req->BuyPrice = buy\_price;  req->SellPrice = sell\_price;  strncpy(req->EnquiryNo, enquiryno, sizeof(DstarApiSystemNoType)); //不填写不指定询价单  req->SeatIndex = 1; **//席位信息中的席位索引,填0不指定席位，由交易服务轮询席位**  req->AccountIndex = m\_LoginInfo.AccountIndex; //登录应答信息中的用户索引  req->UdpAuthCode = m\_LoginInfo.UdpAuthCode; //登录应答信息中的认证码  req->ClientReqId = ClientReqId++;  req->ContractIndex = contractindex; //合约信息中的合约索引  strncpy(req->ContractNo, ContractNo, sizeof(DstarApiContractNoType) - 1);  req->Reference = 1; //要求 >=0  Send(sendbuf, sizeof(DstarApiReqOfferInsertField) + sizeof(DstarApiHead)); //UDP发送 |

下面是发送批量报价单请求的例子：

|  |
| --- |
| char sendbuf[1024] = {0};  int num = 6;  DstarApiHead \*head = NULL;  DstarApiReqOfferInsertField \*req = NULL;  int pos = 0;  for (int i = 0; i < num; i++)  {  head = (DstarApiHead \*)&sendbuf[pos];  head->ProtocolCode = CMD\_API\_Req\_OfferInsert;  head->DataLen = sizeof(DstarApiReqOfferInsertField);  req=(DstarApiReqOfferInsertField \*) &sendbuf[pos + sizeof(DstarApiHead)];  req->BuyOffset = DSTAR\_API\_OFFSET\_OPEN;  req->SellOffset = DSTAR\_API\_OFFSET\_OPEN;  req->OrderQty = qty;  req->BuyPrice = buy\_price;  req->SellPrice = sell\_price;  strncpy(req->EnquiryNo, enquiryno, sizeof(DstarApiSystemNoType)); //不填写不指定询价单  req->SeatIndex = 0; **//席位信息中的席位索引,填0不指定席位，由交易服务轮询席位**  req->AccountIndex = m\_LoginInfo.AccountIndex; //登录应答信息中的用户索引  req->UdpAuthCode = m\_LoginInfo.UdpAuthCode; //登录应答信息中的认证码  req->ClientReqId = ClientReqId++;  req->ContractIndex = contractindex; //合约信息中的合约索引  strncpy(req->ContractNo, ContractNo, sizeof(DstarApiContractNoType) - 1);  req->Reference = 1; //要求 >=0  }  Send(sendbuf, pos); //UDP发送 |

# 11. 协议变更

本节内容只描述协议更改部分，所有协议内容见Api包中的头文件。

表2对应Api变更版本为1.0.0.2/1.0.0.3至1.0.1.0。

表 2 Api协议变更(1.0.0.2/1.0.0.3-1.0.1.0)

|  |  |  |
| --- | --- | --- |
| 协议名称 | 变更类型 | 备注 |
| 合约数据 | 更改 | 添加昨结算、合约到期日字段 |
| 委托数据 | 更改 | 变更委托号类型为整型；添加保证金、报单席位号字段 |
| 报单应答 | 更改 | 变更委托号类型为整型 |
| 报价通知 | 更改 | 变更委托号类型为整型；添加保证金、报单席位号字段 |
| 成交数据 | 更改 | 变更委托号、成交号类型为整型；  添加更新时间字段 |
| 市场状态 | 新增 |  |
| 手续费参数 | 新增 |  |
| 保证金参数 | 新增 |  |
| 浮盈通知 | 新增 |  |
| 报单请求 | 更改 | 添加协议头；变更报单引用类型 |
| 报价请求 | 更改 | 添加协议头；变更报单引用类型 |
| 撤单请求 | 更改 | 添加协议头；变更委托号为整型；添加系统号 |

表3对应Api变更版本为1.0.1.0至1.0.1.1。

表 3 Api协议变更(1.0.1.0-1.0.1.1)

|  |  |  |
| --- | --- | --- |
| 协议名称 | 变更类型 | 备注 |
| 浮盈通知 | 更改 | 添加流号字段 |
| UPD认证请求 | 更改 | 删除协议头字段;使用1字节对齐方式 |
| 报单请求 | 更改 | 删除协议头字段;其他字段顺序进行调整；使用1字节对齐方式 |
| 报价请求 | 更改 | 删除协议头字段;其他字段顺序进行调整；使用1字节对齐方式 |
| 撤单请求 | 更改 | 删除协议头字段;其他字段顺序进行调整；使用1字节对齐方式 |

表4对应Api变更版本为1.0.1.3至1.0.1.4。

表 4 Api协议变更(1.0.1.3-1.0.1.4)

|  |  |  |
| --- | --- | --- |
| 协议名称 | 变更类型 | 备注 |
| 委托数据 | 更改 | “合约编号”修改为“合约编号1”；添加“合约编号2”、“组合号”字段 |
| 成交数据 | 更改 | 添加“组合号”字段 |
| 组合报单请求 | 新增 | 新增“组合报单请求”协议 |

表5对应Api变更版本为1.0.1.4至1.0.1.6。

表 5 Api协议变更(1.0.1.4-1.0.1.6)

|  |  |  |
| --- | --- | --- |
| 协议名称 | 变更类型 | 备注 |
| 登录应答 | 更改 | 添加“交易系统启动时间”、“交易系统启动模式”字段 |
| 席位 | 更改 | 添加“席位状态”字段 |

表6对应Api变更版本为1.0.1.6至1.0.1.7。

表 6 Api协议变更(1.0.1.6-1.0.1.7)

|  |  |  |
| --- | --- | --- |
| 协议名称 | 变更类型 | 备注 |
| 密码修改请求 | 新增 |  |
| 密码修改应答 | 新增 |  |
| 密码修改通知 | 新增 |  |

表7对应Api变更版本为1.0.1.9至1.0.1.10。

表 7 Api协议变更(1.0.1.9-1.0.1.10)

|  |  |  |
| --- | --- | --- |
| 协议名称 | 变更类型 | 备注 |
| 交易权限 | 新增 |  |
| 交易权限删除 | 新增 |  |
| 客户品种  白名单 | 新增 |  |

表8对应Api变更版本为1.0.1.10至1.0.1.11。

表 8 Api协议变更(1.0.1.10-1.0.1.11)

|  |  |  |
| --- | --- | --- |
| 协议名称 | 变更类型 | 备注 |
| 席位 | 更改 | 添加“席位IP”字段 |

表9对应Api变更版本为1.0.1.11至1.0.1.12。

表 9 Api协议变更(1.0.1.11-1.0.1.12)

|  |  |  |
| --- | --- | --- |
| 协议名称 | 变更类型 | 备注 |
| 登录应答 | 更改 | 添加“持仓盈利是否计入可用”字段 |
| 初始化数据查询 | 新增 |  |
| 组合合约数据 | 新增 |  |